Algorithme de Routage Dynamique pour l'Amélioration de la résilience : Ant Colony Routing

1. INTRODUCTION:

Aujourd'hui, dans de nombreux pays, le trafic automobile est un des très grands problèmes surtout avec la croissance du nombre des véhicules et des populations. Si chacun d'entre nous veut choisir le chemin le plus court pour se déplacer d'un endroit D à un autre endroit A, on a déjà des algorithmes pour trouver ce plus court chemin mais le problème est que des embouteillages risquent d'être créés sur certaines routes, dû au nombre de voitures circulant sur ces routes et à leurs capacités limitées; ou à cause d'autres événements aléatoires, comme les phénomènes météorologiques (brume, brouillard, crachin, pluie, neige, grêle et orage) ou encore les accidents routiers qui peuvent aussi engendrer des congestions sur le réseau. Dans les situations où des congestions se créent sur certaines routes pour une ou plusieurs raisons, des agents des centres de gestion du trafic doivent intervenir afin de diffuser l'information, et rediriger les voitures qui arrivent sur une route congestionnée pour éviter une congestion plus grave du trafic. Le but de cette feuille de recherche est de trouver un algorithme adapté à un cadre de travail de simulation MaTSim, capable de prendre en compte ces encombrements qui peuvent arriver et d'assurer une bonne fluidité sur tout le réseau.

2. TRAVAUX LIÉES:

Pour trouver le plus court chemin entre deux points dans un réseau routier, il existe plusieurs algorithmes. Le plus connu de ces algorithmes est celui de Dijkstra [1] , ou sa version amélioré en terme de temps de calcul l'algorithme de A* [2]. Ces deux derniers algorithmes ne prennent pas en compte les changements dynamiques et les congestions qui peuvent être créés sur le réseau, ils font partie des algorithmes de routage statique.

Pour résoudre notre problème, nous avons besoin d'un algorithme de routage qui s'adapte aux changements dynamiques qui vont se produire sur le réseau. On peut identifier un bon candidat si on observe le comportement des fourmis. Pourtant une fourmi est juste un simple insecte avec un petit champ de vision, mais plusieurs fourmis ensemble vont former ce qu'on appelle une intelligence d'essaim ou bien "Swarm Intelligence" en anglais, ou chaque fourmi va déposer de la phéromone* sur le sol, là où elle passe, afin de créer un système de signalement, qui va leur permettre de trouver le meilleur chemin de leurs nid vers la source de nourriture et retourner à leur nid.Les fourmis arrivent également à adapter leurs chemins en fonction des changements dynamiques.

Au tout début, les chercheurs se sont inspirés de ce comportement de fourmis qui a été observé en premier par des biologistes en 1989. Pour résoudre des problèmes d'optimisation, ils ont inventé ce qu'on appelle Les algorithmes de colonies de fourmis ou bien en anglais(**A**nt **C**olony **O**ptimization), qui servent à construire une famille de métaheuristiques d'optimisation.

Dès que les premiers chercheurs ont réussi à appliquer l'algorithme de colonies de fourmis sur des problèmes d'optimisation NP difficile, comme dans [3] où on applique l'algorithme pour résoudre le problème du voyageur de commerce. Cet algorithme de colonie de fourmis à commencé à attirer de plus en plus de chercheurs qui essaient et qui ont déjà réussi à l'appliquer sur différents problèmes jusqu'à aujourd'hui.

On trouve des variantes de l'ACO pour la résolution des problèmes d'équilibre de charge dans les réseaux téléphoniques comme dans [4] ou l'algorithme AntNet décrit dans [5] qui sert à résoudre le problème de routage des paquets dans un réseau. Également **AntHocNet** [6] et **HopNet** [7] sont deux autres applications de l'ACO pour

des problèmes de routage dans des réseaux sans fil ou plus récemment les travaux réalisés dans [8] afin de résoudre le problème de routage et l'allocation de spectre sur des réseaux avec une bande passante flexible.

Il existe aussi d'autres applications de l'algorithme de colonie de fourmis dans le domaine de la classification d'image[9] [10]. On peut citer d'autres applications et approches à base d'algorithmes de colonies de fourmis, mais on va s'intéresser plus précisément à ceux proches de notre problématique. Typiquement les travaux de Bogdan Tatomir et Leon Rothkrantz [11], où ils ont essayé de développer un système de routage dynamique qui s'adapte bien aux problèmes qui peuvent se produire sur le réseau; ainsi que les travaux réalisés dans [12], ou les auteurs proposent un nouvel algorithme de routage dynamique, pour trouver un routage optimal en termes de temps total de voyage et en essayant de ne pas dépasser un certain nombre de véhicules sur quelques routes sensibles comme les routes à côté des hôpitaux par exemple, pour une population donnée, avec différentes origines et destinations en passant par deux étapes principales, le filtrage du réseau afin d'exclure certaines routes pour diminuer le temps de calcul puis l'utilisation d'un système de prédiction basé sur la phéromone. Ils ont introduit un deuxième type de phéromone dite répulsive pour éviter les bouchons et ils ont également séparé la phéromone en plusieurs phéromones colorées, afin d'attribuer une phéromone colorée pour chaque destination différente.

Ce dernier algorithme de routage [12] a montré une bonne efficacité et a permis d'atteindre un bon équilibre entre les performances de contrôle et la vitesse de calcul notamment grâce à l'étape de filtrage du réseau. Cependant cet algorithme ne répond pas exactement à ce qu' on cherche à faire, vu qu' il travaille avec un système d'optimisation en ligne avec un système de prédiction, alors que dans notre cas nous avons besoin d'un système fonctionnel en temps réel capable de s'adapter aux changements qui peuvent se produire sur le réseau.

Motivés par ce problème et les contraintes à respecter nous allons dans la suite de cette feuille de recherche proposer deux versions différentes d'algorithmes de routage dynamique, une première pour les cas où toutes les personnes de la population ont la même destination et une deuxième pour les cas où les personnes de la population ont différentes destinations. Avec des variantes pour chaque algorithme, et à la fin nous allons tester les deux versions de notre algorithme et leurs différentes variations afin de les évaluer et de les comparer également. Mais avant ça nous allons tout d'abord introduire notre problème, le formaliser et présenter le cadre du travail.

Modélisation du problème

Le but de cette feuille de recherche est d'implémenter un algorithme de routage dynamique basé sur l'algorithme de fourmis, qui est capable de satisfaire les contraintes suivantes:

- Optimiser le temps total de voyage de toute la population.
- Détecter les embouteillages et s'auto adapter en fonction de ces congestions créées et les gérer en redirigeant les voitures qui essaient d'entrer dans des routes congestionnées vers d'autres routes qu'ils ne sont pas.

Par la suite nous allons modéliser notre réseau routier par un graphe, composé d'un ensemble de noeuds$N$et un ensemble d'arcs A, chaque arc $a \in A$est unidirectionnel, il a une capacité limitéec_a, une longueur l_a. On notera $O(n)$, l'ensemble des arcs sortant d'un noeud $n$et $I(n)$l'ensemble des arcs entrant dans le noeud

ACR Algorithme pour une seule destination

Dans cette première version de l'ACR nous allons considérer que toutes les voitures de la population ont la même destination. Elles vont être considérées comme des fourmis et notre algorithme sera basé sur une modélisation de la phéromone. On va donc associer à chaque arc une valeur de phéromone τ_a, initialisée au début de l'algorithme a une valeur $\tau_0 > 0$ pour l'ensemble des arcs du graphe A. Cette valeur de

phéromone nous indique à quel point cet arc va attirer des voitures à le traverser. À chaque fois q'une fourmi v arrive sur un nouveau noeud n, tant que $n \neq t_f$ elle va essayer de choisir un arc a_r parmi $O(n)$,d'une façon probabiliste qu'on va expliquer dans la section suivante.

Les voitures vont partir d'un noeud précis s, et il vont essayer tous d'atteindre le nœud d. À chaque fois qu'une voiture v se trouve sur un noeud n, elle va essayer de choisir un arc a_r parmi $O(n)$,d'une manière probabiliste.

Calcul des probabilités

La fourmi va commencer d'abord par calculer une probabilité $P(a_i)$ de choisir un arc a_i parmi $O(n)$, ou n est le sommet où elle se trouve, avec la formule suivante :

$$P(a_i|O(n)) = \frac{(\tau_{a_i})^{\alpha} * (\frac{1}{max\{Distance(S(a_i),t),0\}})^{\beta}}{\sum_{a_j \in O(n)} (\tau_{a_j})^{\alpha} * (\frac{1}{max\{Distance(S(a_i),t),0\}})^{\beta}} \quad (1)$$

Avec τ_{a_i} la valeur de phéromone sur l'arc a_i, D(n',t) la distance entre le nœud d'arrivée t et le le nœud successeur de a_i noté S(a_i).α et β sont deux paramètres qui déterminent l'importance relative de la valeur de phéromone et de la distance dans le calcul des probabilités.

Choix de la prochaine route:

Une fois que la fourmi a calculé un ensemble de probabilités P_n pour l'ensemble $O(n)$ d'un noeud n, la voiture va choisir un seul élément parmi $O(n)$, en passant par une variante simplifiée de la méthode de sélection probabiliste appelée "**Roulette Wheel"**(: une méthode de sélection utilisée la plupart du temps pour des algorithmes génétiques comme ce qui a été fait dans [13]). Dans notre version simplifiée on va générer un nombre réel aléatoire $r \in [0, 1]$. Après ça nous allons diviser une droite graduée en m=$|P(n)|$ sous parties; où la première sous partie va être sur l'intervalle $[0, P_0]$ et où P_0 est la première probabilité de l'ensemble P(n), la deuxième sous partie va se trouver dans l'intervalle $[P_0, P_0 + P_1]$ et , ainsi de suite, jusqu'à la dernière sous partie qui va se retrouver dans l'intervalle $\left[\sum_{i=0}^{i=m-1} Pi\ , 1\right]$. Finalement, pour choisir le prochain arc à traverser, on va regarder sur quelle sous partie de la droite graduée se trouve la valeur de r, et cela va correspondre à l'indice de l'arc à choisir parmi l'ensemble $O(n)$. (voir Figure 01 dans l'annexe)
Avec ce système de choix actuel et les valeurs de phéromones identiques sur tous les arcs du réseau, nos voitures(fourmis) vont choisir d'une manière aléatoire à chaque fois le nœud qui va les rapprocher le plus vers leurs destinations. Elles peuvent parfois prendre un chemin qui n'est pas forcément l'optimum en termes de temps de voyage total. Pour s'assurer que les fourmis convergent vers un chemin optimal en termes de temps de voyage, à chaque fois qu'une fourmi f, arrive à sa destination t, en passant par un ensemble d'arcs appelé $C(v)$, on met à jour les valeurs de phéromones de tous les arcs du graphe avec la formule suivante :

$$\left\{\tau_{a_i} = \tau_i * (1 - \rho_{evap}) + \frac{1}{TTS_f} \; si \; a_i \in C(v)\right\}$$
$$\left\{\tau_{a_i} = \tau_i * (1 - \rho_{evap}) \qquad si \; a_i \notin C(v)\right\} \quad (2)$$

Où $\rho_{evap} \leq 1$ représente le taux d'évaporation, un paramètre fixé au début de l'algorithme, TTS_f représente le temps total de voyage de la fourmi f. Avec une telle mise à jour de la valeur de phéromone, à chaque arrivée d'une fourmi, après un certain temps toutes les fourmis vont converger vers le chemin optimal en termes de temps total de voyage, même si des risques de tension peuvent se créer sur certains arcs de ce chemin optimum.

Pour éviter ce phénomène nous allons utiliser un deuxième type de phéromone, dite répulsive, qui va nous servir à repousser les fourmis des arcs congestionnés, afin de ne pas aggraver la tension sur certains liens déjà congestionnés ou bien pour limiter le trafic sur certaines routes du réseau. À chaque fois qu'une fourmi f sort d'un arc a, elle va déposer une quantité de phéromone répulsive sur ce dernier arc, correspondante au taux de congestion sur ce lien dans le but de laisser aux autres fourmis une information concernant la qualité du trafic sur cet arc.

Dans cette première sous partie de l'algorithme, où on travaille sur des fourmis avec une seule destination, la phéromone répulsive va être intégrée dans la phéromone globale. C'est à dire qu' à chaque fois qu'une fourmi f traverse un nouveau noeud n, en passant par un arc a_l, elle va mettre à jour la valeur de la phéromone attractive de ce dernier τ_{a_l} selon la formule ci-dessous :

$$\tau_{a_l} = \tau_{a_l} + RF(f, a_l) \qquad si \;\; 0 \preccurlyeq \tau_{a_l} + RF(f, a_l)$$
$$\tau_{a_l} = \frac{\tau_0^{\,2}}{\tau_{a_l} + RF(f, a_l)} \qquad si \;\; \tau_{a_l} + RF(f, a_l) \prec -1 \qquad (3)$$
$$\tau_{a_l} = \tau_0 * (1 + RF(f, a_l)) \quad si \;\; \tau_{a_l} + RF(f, a_l) \prec 0$$

Avec τ_{a_l} la valeur de phéromone sur l'arc a_l, a_0 la valeur de phéromone initiale et $RF(f, a_l)$ une fonction pour calculer la valeur de phéromone répulsive (RF) à déposer par la fourmi f sur l'arc a_l au moment de sa sortie de ce dernier (a_l).

Dans la partie implémentation de l'algorithme, nous avons implémenté deux différentes versions de RF:

- Une première version qui dépend du temps passé sur l'arc, la vitesse limite de l'arc et sa longueur.
- Une deuxième version qui dépend du nombre des fourmis sur l'arc au moment où la fourmi sort de cet arc et sa capacité.

Les deux versions peuvent être également paramétré par un paramètre δ, avec $0 <= \delta$ qui permet d'indiquer à partir de quelle qualité de trafic sur cet arc, il sera considéré comme congestionné. Ce paramètre permet aussi d'identifier les arcs les plus sensibles (ceux avec une valeur δ proche de 0), et d'assurer de ne pas les encombrer .

Vous trouverez une comparaison entre ces deux versions de $RF(a)$, dans la partie portant sur l'analyse des résultats.

Avec un tel algorithme, les fourmis sont capables d'aller d'un noeud de départ vers un noeud d'arrivée, les congestions qui vont se créer sur certains arcs du chemin optimal, vont être gérées grâce à la phéromone répulsive qui va diminuer la valeur de phéromone sur les arcs congestionnés et donc inciter les fourmis à ne pas prendre des chemins congestionnés.

Un arc congestionné va voir la valeur de sa phéromone décrémentée, mais à l'arrivée des fourmis sa valeur va être incrémentée encore une fois, ce qui laisse la possibilité de choisir ce même arc une autre fois qu'il n' est plus congestionné et ne pas l'éliminer du réseau définitivement. Donc jusqu' ici, on peut dire que ACO pour une seule destination est fonctionnel, comme on va le tester et l'évaluer dans la partie implémentation.

ACR Algorithme pour différentes destinations

Si on essaie d'appliquer la première version de ACR, pour une population avec le même noeud de départ et différentes destinations, nous allons voir que des fourmis avec une destination t , vont être attirées par la phéromone déposée par des fourmis avec une destination différente t'. Donc il vont suivre un chemin qui n'est pas du tout adapté à leur destination.

Afin de résoudre ce problème nous allons considérer une phéromone attractive par destination, on peut les distinguer par exemple par des couleurs. Chaque fourmi sera attirée par la phéromone correspondant à sa destination. Les valeurs de cette phéromone seront initialisées de la même façon que pour **l'ACR pour une seule destination**, c'est à dire on va attribuer au début de l'algorithme $|D|$ valeurs de phéromones initialisées toutes à une constante positive τ_0, Détant l'ensemble des différentes destinations.

Les fourmis vont choisir le prochain arc avec le même principe utilisé dans l'ACR pour une seule destination formule (1) , avec une petite différence dans le calcul des probabilités. La fourmi va calculer la probabilité de chaque arc en fonction de la valeur de phéromone correspondant à sa destination. La formule pour calculer les probabilités de chaque lien pour aller à t_f devient:

$$P(a_i|O(n)) = \frac{(\tau_{a_i t_f})^{\alpha} * (max\{ Distance(S(a_i), t_f), 0\})^{\beta}}{\sum_{a_j \in O(n)} (\tau_{a_j t_f})^{\alpha} * (max\{ Distance(S(a_j), t_f), 0\})^{\beta}} \quad (4)$$

À chaque fois qu'une fourmi f arrive à sa destination $t1$, elle va modifier les valeurs de phéromones correspondantes à sa destination avec le même procédé que dans l'ACR pour une seule destination:

$$\left\{ \tau_{t1_{a_i}} = \tau_{t1_{a_i}} * (1 - \rho_{evap}) + \frac{1}{TTS_f} \; si \; a_i \in C(v) \right\} \quad (5)$$

$$\left\{ \tau_{t1_{a_i}} = \tau_{t1_{a_i}} * (1 - \rho_{evap}) \qquad si \; a_i \notin C(v) \right\}$$

Où $\tau_{t1_{a_i}}$ est la valeur de phéromone pour la destination $t1$, sur l'arc ai,ρ_{evap} le taux d'évaporation et TTS_V est l'inverse du temps total de voyage de la fourmi.

Pour éviter les congestions, on va travailler avec le même principe de phéromone répulsive que celui utilisé dans l'ACR pour une seule destination. On va réutiliser la même formule(3).La différence est juste que pour cette version la fourmi va modifier la valeur de phéromone de l'arc qu'elle vient de quitter liée à sa destination, ainsi que toutes les autres valeurs de phéromone liées à d'autres destinations, donc la nouvelle formule de mise à jour des valeurs de phéromones répulsives devient

$$\tau_{t_i a_l} = \tau_{t_i a_l} + RF(f, a_l) \qquad si \;\; 0 \preccurlyeq \tau_{t_i a_l} + RF(f, a_l)$$

$$\forall t_i \in D \qquad \tau_{t_i a_l} = \frac{\tau_0^{\,2}}{\tau_{t_i a_l} + RF(f, a_l)} \qquad si \;\; \tau_{\tau_{t_i a_l}} + RF(f, a_l) \prec - 1 \qquad (6)$$

$$\tau_{t_i a_l} = \tau_0 * (1 + RF(f, a_l)) \quad si \;\; \tau_{\tau_{t_i a_l}} + RF(f, a_l) \prec 0$$

Mise en oeuvre dans l'environnement Matsim

Nous avons implémenté notre ACR afin de l'intégrer dans MATSim. MATSim est un cadre de travail (**framework** en anglais),open-source pour la mise en œuvre de simulations de transport à grande échelle basées sur des agents. Il se compose de plusieurs modules qui peuvent être combinés ou utilisés de manière autonome. Les modules peuvent être remplacés par des implémentations personnalisées pour tester des aspects particuliers de notre propre travail.

MATSim offre un cadre pour la modélisation de la demande, la simulation de la mobilité basée sur des agents (simulation du flux de trafic), la planification, un contrôleur pour exécuter des simulations de manière itérative ainsi que des méthodes pour analyser les résultats générés par les modules.cf Figure1 Matsim Loop

Notre algorithme de routage va intervenir pendant l'exécution de la simulation, on va donc l'intégrer au module principal d'exécution mobism, pour jouer le rôle d'un planificateur pour nos agents, ou bien ce qu'on appelle dans MATSim within-day-replaning. cf Figure 2 Boucle Matsim avec l'extension within day replaning.

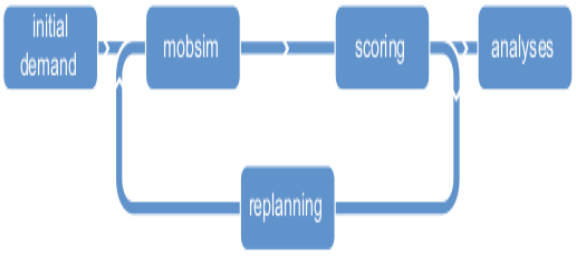


Figure1 : Boucle principale de MaTSim

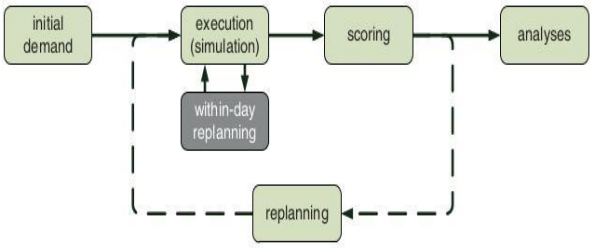


Figure2 : Boucle Matsim avec l'extension within-day-replaning

Les réseaux utilisés dans MATSim[REF] ressemblent au graphe introduit dans la partie (X). La particularité est que les routes sur les réseaux de MATsim ont un attribut de plus que les arcs du graphe(une longueur$L$, vitesse maximum MS et une capacité C), c'est le nombre de voies annoté par un N_{lanes} mais il reste toujours unidirectionnel comme les arcs de notre graphe.(cf Figure 2 dans l'annexe)

Nous avons pu récupérer les différentes données requises par notre algorithme au cours de l'exécution grâce aux différents événements qui se produisent pendant la simulation et des events handler attachés au contrôle pour les traiter et stocker les informations dans des structures de données adaptées.(cf Figure 3 structure de l'implémentation.)

Pour la visualisation des données et l'analyse nous avons utilisé l'outil https://simunto.com/via/Via de Simunto, conçu pour lire les fichiers en sortie de Matsim, les visualiser et de faire de l'analyse sur ces données.

Nous avons également bien défini les différentes versions de la fonction $RF(l)$, qui calcule le taux de phéromone à déposer par chaque voiture, au moment de la sortie d'un lien l, qui va dépendre d'un seuil de sensibilité δ qu'on peut généraliser pour tous les liens du réseau, ou bien utiliser différentes valeurs pour chaque lien.

- La première version de $RF(l)$ qui dépend du temps passé par la voiture$v$ sur le lien $l$est définie avec la formule suivante:

$$RFV(v,l,\delta) = \frac{-Tt_{v,l}}{\frac{l_l}{MS_l}*(1+\delta)} - 1$$

- La deuxième version de $RF(l)$, qui dépend du nombre de véhicules N sur le lien au moment de la sortie de la voiture v de ce lien l, et une estimation de la capacité du lien est avec l'expression suivante :

$$RFN(v,l,\delta) = -\frac{N}{\delta*\frac{N_{lanes}*C_l}{7.5}}+1$$

L'implémentation codée peut être utilisée pour une population avec une unique destination ou différentes destinations. La version de phéromone répulsive peut être sélectionnée au début ou désactivée:(On peut tester le comportement de l'algorithme juste avec la phéromone attractive)

Analyse des résultats et comparaison entre les différentes variantes

Dans cette partie nous allons tester les différentes versions de notre algorithme, nous allons évaluer les performances de chaque algorithme par rapport à deux métriques:

- Le temps total de voyage moyen TTS_{moyen}.
- Vitesse relative moyenne $VR_{moyenne}$ de l'ensemble des liens traversés au moins une fois pendant la simulation $\mathcal{L}$. La vitesse relative moyenne permet d'estimer le taux de fluidité sur le réseau. La fluidité du réseau et la vitesse relative moyenne sont inversement corrélées avec $1 \leq VR_{moyenne}$ pour n'importe quelle simulation. La formule utilisée pour calculer la vitesse relative sur un lien $l$est la suivante :

$$VR_l = \frac{\sum_i^{|Ttl_l|} Ttl_l[i]}{|Ttl_l|*\frac{L_l}{VM_l}}$$

où Ttl_lest l'ensemble des temps voyages sur le lien l, $L_l$la longueur du lien et VM_lla vitesse limitée sur ce lien. Donc la formule de $VR_{moyenne}$ donne :

$$VR_{moyenne} = \frac{\sum_{l\in\mathcal{L}} VR_l}{|\mathcal{L}|}$$

On va également montrer à chaque fois les valeurs de phéromones sur chaque après la fin de la simulation.

Au début, nous avons commencé par tester notre algorithme sur une population de 500 voitures, qui partent tous au même moment du nœud 5 vers le nœud 19. (cf Figure 4 dans l'annexe) Et nous avons adopté les paramètres ci-dessous pour notre algorithme
($\alpha = 3$, $\beta = 3$, $\rho_{evap} = 0.001$, $\tau_0 = 0.2$, $\delta = 0.1$).

Le tableau ci-dessous(Figure 3) présente les valeurs moyennes de 10 exécutions de la même simulation décrite un peu plus haut. Après cette première série de tests, on peut voir l'effet des deux versions différentes de la phéromone répulsive sur la fluidité du réseau. Ainsi, on a pu constater que la version de phéromone répulsive qui dépend du nombre de personnes RFN, montre de meilleurs résultats en terme de fluidité du réseau et temps de voyage moyen par rapport à la version qui dépend du temps passé sur le lien RFV.

La version utilisée	Sans phéromone répulsif	Phéromone répulsif en fonction du temps passé	Phéromone répulsif en fonction du nombre de personne
$TTS_{moyen}(min)$	108	124	112
$VR_{moyenne}$	2.47	1.64	1.33

Figure 3: Performances moyenne de 10 exécutions de la même simulation

La population guidée par l'ACR sans phéromone répulsive a réalisé un meilleur temps de voyage moyen par rapport aux autres versions. Grâce à la convergence des valeurs de phéromone attractives sur le plus court chemin en termes de temps de voyage tout au long de la simulation.
La figure 5 dans l'annexe exhibe la phéromone attractive de chaque lien du réseau au début et à la fin d'une simulation sans phéromone répulsive.

Comparaison entre les deux fonctions de phéromones répulsive

Pour mieux comprendre la différence entre les différentes versions nous avons observé l'évolution de la phéromone sur le lien 12 et 13, les deux liens les plus susceptibles d'être congestionnés en premier pour notre scénario. (Voir les figures 6 et 7 dans l'annexe)

La valeur de phéromone sur le lien 12 n'évolue pas de la même façon pour les deux variantes. On peut voir sur les deux figures précédentes, que la RFNnous assure une détection des congestions plus rapide par rapport à la RFV, Également la reprise du trafic sur un lien déjà congestionné ou la qualité du trafic vient de s'améliorer se fait plus rapidement avec la RFN.

La figure 8 dans l'annexe montre deux situations o$\grave{u}$ la RFN sera meilleure que la RFVpour la détection et la reprise du trafic.

Dans la situation 8.1 représentée sur la figure W, la RFN() détecte plus rapidement la congestion qui peut occuper sur le lien, si on suppose qu'il n'avait pas de voitures avant sur ce lien. À la sortie de la première voiture, la RFN() va retourner une valeur plus importante que celle de RFV().
Pour la même situation, maintenant au moment de la sortie de la dernière voiture qui a traversé ce lien, la RFN() va retourner une valeur inférieure à celle retournée par la RFV(), ce qui signifie une reprise de trafic plus rapide.

Pour conclure cette partie, les performances montrées par les différentes versions sont dépendantes des paramètres de notre algorithme fixés au début et également de la taille de la population et sa demande.

Evaluation et analyse de l'ACR pour plusieurs destinations:

Nous allons reprendre les mêmes paramètres et populations utilisés dans la partie d'évaluation d'ACR pour une seule destination. La différence est que la moitié de la population va essayer d'aller au noeud 22 et l'autre moitié va garder le noeud 19 comme destination.

La version utilisée	Sans phéromone répulsif	RFV	RFN
$TTS_{moyen}(min)$	67	81	77
$VR_{moyenne}$	2.32	2.04	1.78

Figure : Performances de l'ACR pour différentes destinations

Nous avons constaté par le biais des séries de simulation avec l'ACR multiples destinations, que la FRN est meilleur en matières de $VR_{moyenne}$ par rapport aux deux autres versions peu importe la complexité du problème. Ainsi que la convergence des valeurs de phéromones des simulations avec l'ACR sans phéromone répulsive est approuvée même pour des populations à destination multiples

La vitesse relative moyenne $VR_{moyenne}$, des simulations multi destinations avec phéromone répulsive, RFV ou RFN est plus aiguë que celle des simulations à destination unique.

Cette augmentation est due à la vitesse relative moyenne des liens prédécesseurs d'une destination. Ces derniers liens seront toujours congestionnés même si on utilise une phéromone répulsive et on prend la même valeur pour α et β. La probabilité de choisir ces liens comme prochaine route reste toujours élevée à cause de l'inverse de la distance qui sera égale à 1. (voir formules (1) et (4)) .

Conclusion:

Dans cette feuille de recherche, nous avons développé un algorithme de routage dynamique ACR basé sur l'algorithme des fourmis ACO. L'ACR vient avec deux versions différentes et permet de détecter et d'éviter les congestions sur le réseau. Nous l'avons intégré à MaTSim afin qu'on puisse le tester sur des simulations.

Grâce aux tests que nous avons effectués, on peut voir qu'une valeur de phéromone dépendante du nombre de voitures sur le lien est meilleure qu'une version basée sur les vitesses.

Finalement, notre algorithme a montré de bonnes performances pour la détection et la gestion des congestions sur la plupart des liens du réseau, mais un peu moins sur les liens précédents des nœuds destinations.

Une autre évolution possible pour notre algorithme dans le futur, ça va être de développer une version distribuée. Où on peut affecter un ACR pour sous régions d'un réseau plus grand.

RÉFÉRENCES:

[1] Yong Deng, Yuxin Chen, Yajuan Zhang, Sankaran Mahadevan,Fuzzy Dijkstra algorithm forshortest path problem under uncertain environment,Applied Soft Computing,Volume 12, Issue 3,2012,Pages 1231-1237,ISSN 1568-4946,

[2] I. Chabini and S. Lan, Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks, IEEE Transactions on Intelligent Transportation Systems,Vol. 3, No. 1, 2002, pp. 60-74.*

[3] Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. Biosystems, 43(2), 73–81. doi:10.1016/s0303-2647(97)01708-5

[4] R. Schoonderwoerd, O.E. Holland and J.L. Bruten and L.J.M.Rothkrantz, Ant-Based Load Balancing in Telecommunications Networks, Adaptive Behavior, vol. 2, 1996, pp. 169-207.

[5] G. Di Caro and M. Dorigo, AntNet: distributed stigmergetic control for communication networks, Journal of Artificial Intelligence Research (JAIR), vol. 9, 1998, pp. 317-365.

[6] G. Di Caro, F. Ducatelle L.M. Gambardella, AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks,European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking, Vol. 16, No. 5, 2005.

[7] Wang, J., Osagie, E., Thulasiraman, P., & Thulasiram, R. K. (2009). HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. Ad Hoc Networks, 7(4), 690–705.

[8] Wang, Y., Zhang, J., Zhao, Y., Wang, J., & Gu, W. (2013). ACO-based routing and spectrum allocation in flexible bandwidth networks

[9] Chen, B., Chen, L., & Chen, Y. (2013). Efficient ant colony optimization for image feature selection. Signal Processing, 93(6), 1566–1576.

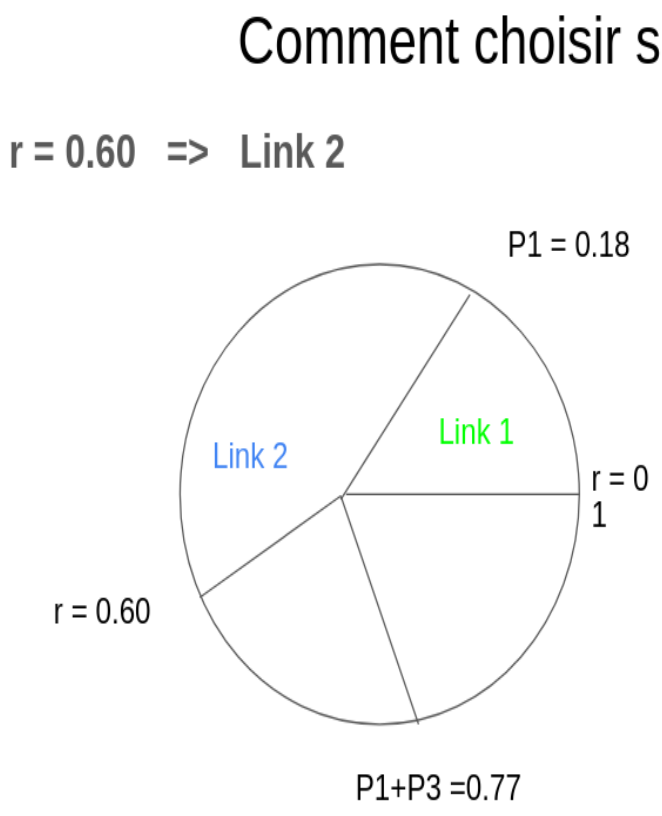
[10] Wang, M., Wan, Y., Ye, Z., & Lai, X. (2017). Remote sensing image classification based on the optimal support vector machine and modified binary coded ant colony optimization algorithm.

[11]B. Tatomir and L. Rothkrantz, "Hierarchical Routing in Traffic Using Swarm-Intelligence," 2006 IEEE Intelligent Transportation Systems Conference, 2006, pp. 230-235.

[12] Zhe Cong, Bart De Schutter, Robert Babuška,Ant Colony Routing algorithm for freeway networks,Transportation Research Part C: Emerging Technologies,Volume 37,2013,Pages 1-19,ISSN 0968-090X.

[13] Adam Lipowski, Dorota Lipowska,Roulette-wheel selection via stochastic acceptance,Physica A: Statistical Mechanics and its Applications,Volume 391, Issue 6,2012,Pages 2193-2196,ISSN 0378-4371.

ANNEXE



Liens en sorties	Probabilités
Link 1	**0.18**
Link 2	**0.59**
Link 3	**0.3**
Sommes	1.0

Figure 1

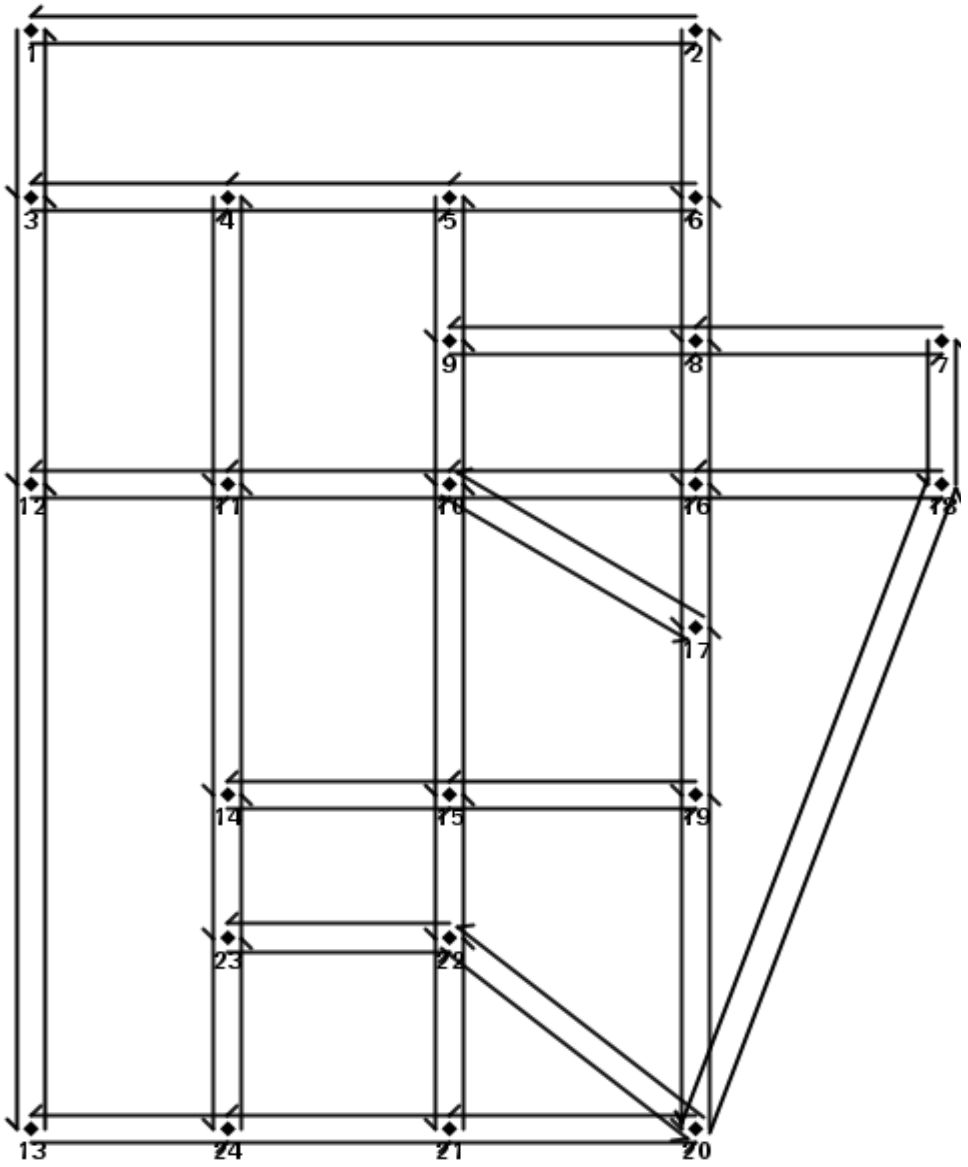


Figure 2: Le réseau de Matsim

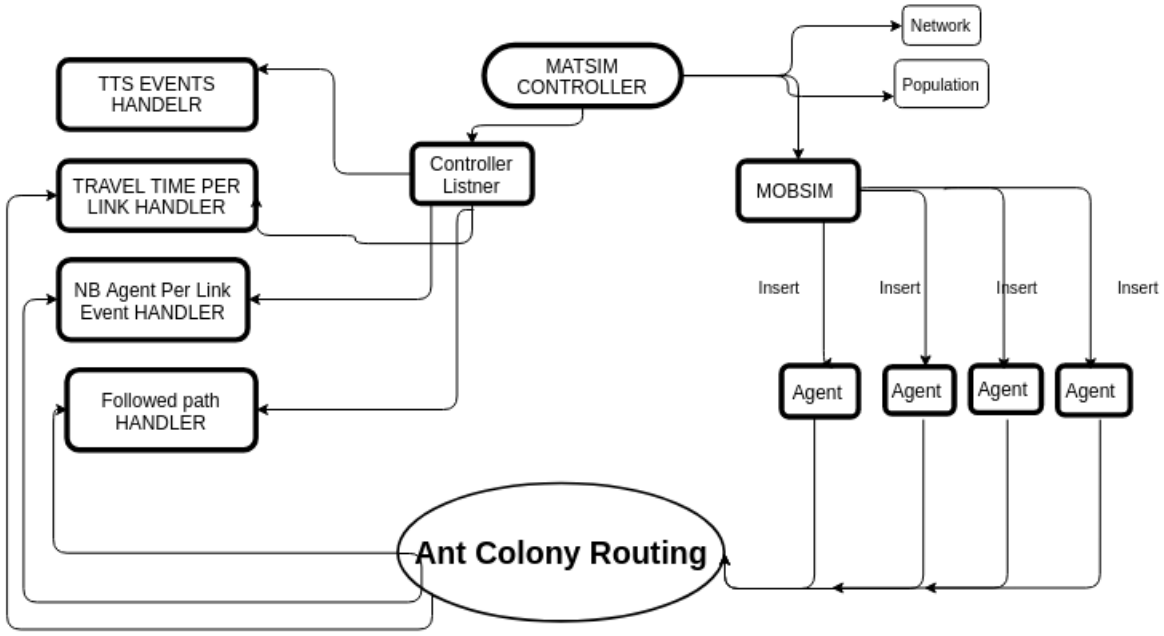


Figure 3 :Structure de l'implémentation

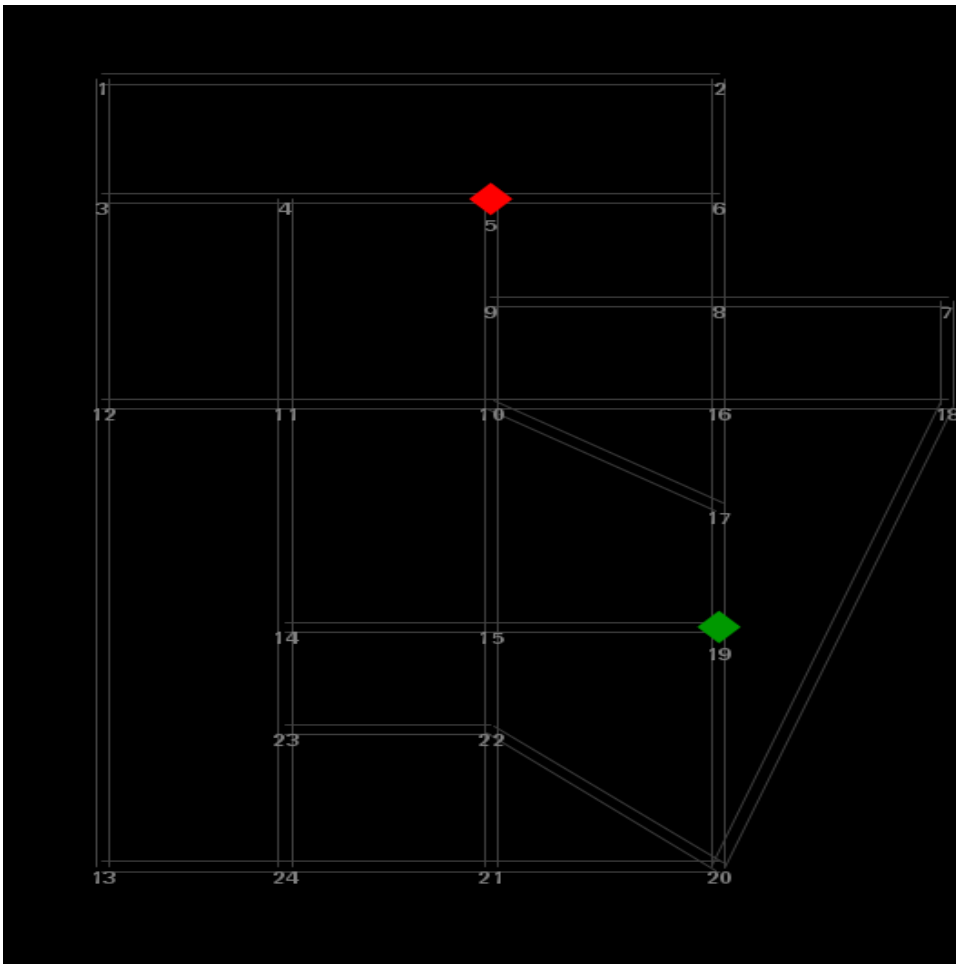
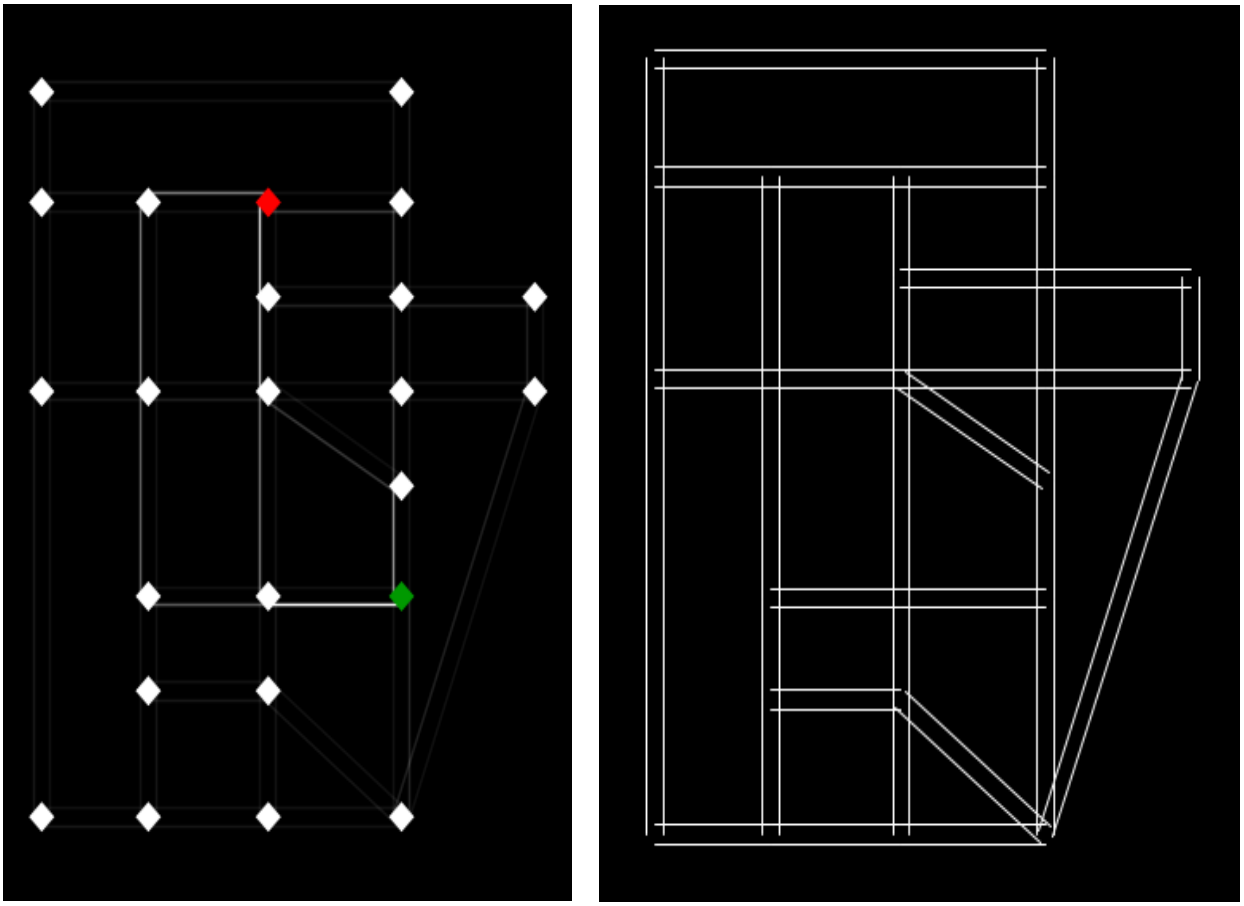


Figure 4 :Noeud de départ et d'arrivée



Phéromones avant convergence

Phéromones après convergence

Figure 5

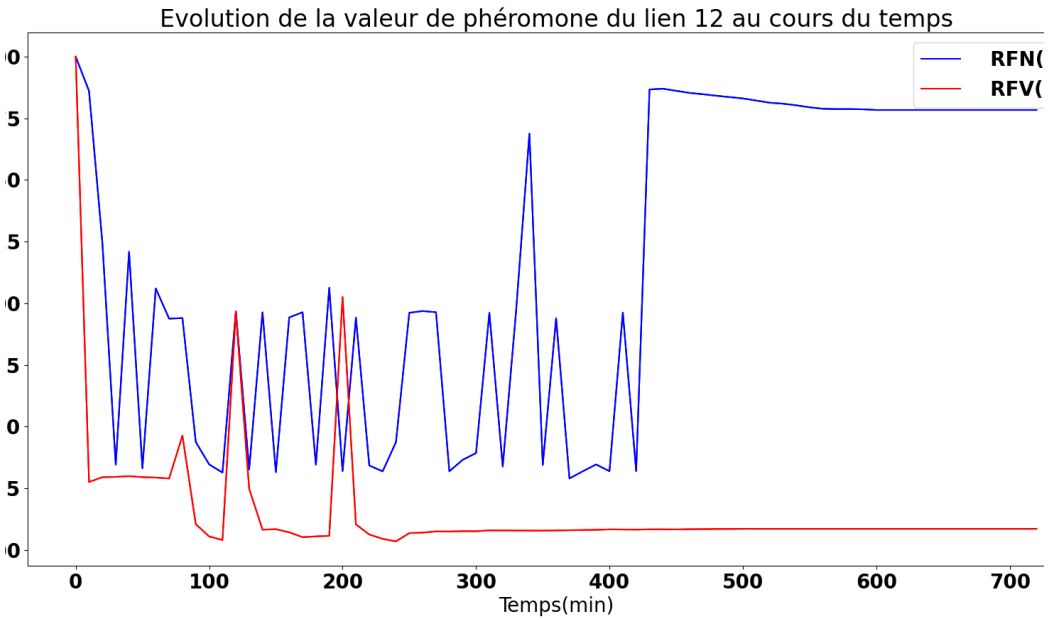


Figure 6 : Evolution de la phéromone sur le lien 12

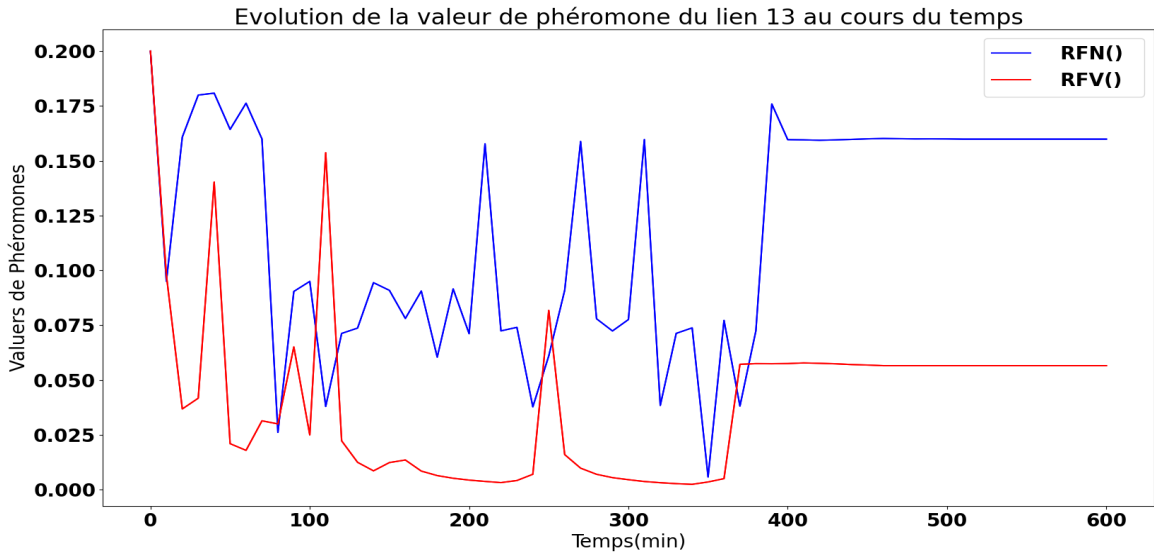


Figure 7 : Evolution de la phéromone sur le lien 13

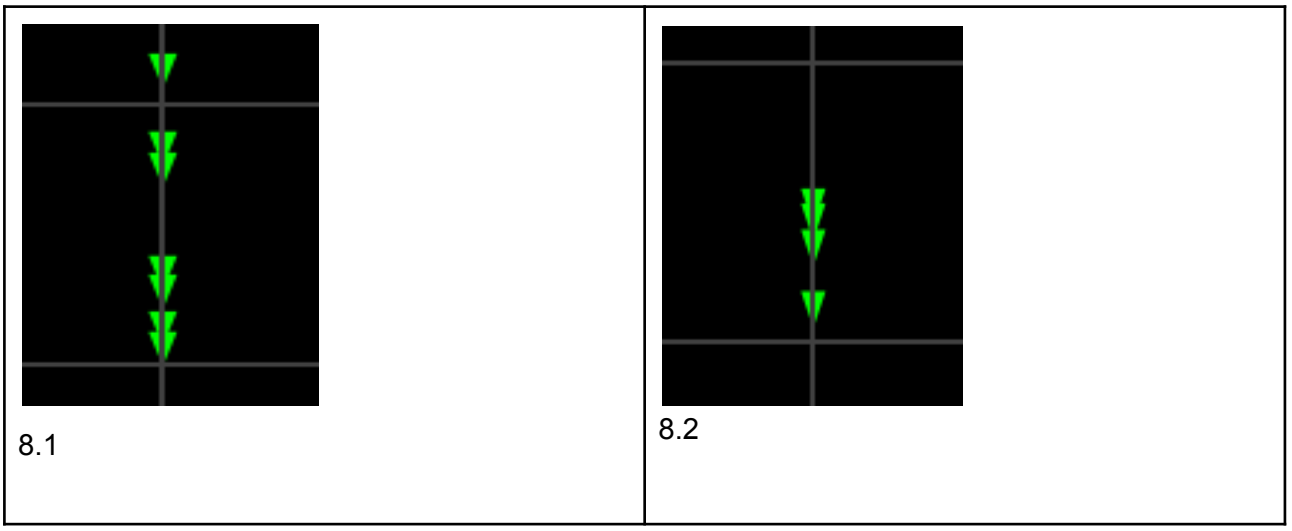


Figure 8: Deux différentes situations d'excellence pour la RFN par rapport à RFV